2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

WWN: Integration with Coarse-to-fine, Supervised and Reinforcement Learning

Zejia Zheng, Juyang Weng, and Zhengyou Zhang

***Abstract*—The cost of autonomous development is substantial. Although supervised learning is effective, the cost demand on teachers is often too high to be constantly applied. Reinforcement learning can take advantage of physical reality due to environmental feedback and inspections. Information required in reinforcement learning is not as specific as is required in supervised learning. Integration theories, methods, and analysis of these two learning strategies are still rare in the literature although such integration has been well known in the animal kingdom. Based on our prior work on a general purpose framework called Developmental Network and its embodiment Where-What-Network, we present our theory, method, and analysis for integration of supervised learning and reinforcement learning in this paper. Different from all other known work on reinforcement learning, this DN framework uses fully emergent representation to avoid the brittleness and task-specific representations. Central in the integration is not just to provide a freedom for the teacher to choose the mode of learning, which is necessary especially when the physical non-living world is an implicit teacher, but the mechanism of scaffolding. In our experiment the scaffolding is reflected by allowing the location motor(LM) neurons to gradually refine representation through splitting(mitosis) in a coarse to fine scheme. We report our experimental work in a very challenging learning setting: both object and backgrounds are unknown(cluttered settings) and concepts(e.g. location and type) emerge from agent-environment interactions, instead of rigidly handcrafted.**

I. Introduction

Integrated learning is prevalent in animal kingdom. Parents supervise their baby's basic motor skills while the baby practice and refine his action based on environmental feedbacks. On the other hand, computational models use separate modules in different modes of learning, making the representation and learned skills hard to transfer between modes.

Our prior work modeled effects of serotonin and dopamine on motor neurons where serotonin discourages firing of the motor neurons while dopamine encourages firing [9]. Our work last year expanded the effect of these two neuromodulators to all neurons in internal brain area of the developmental network [17].

In this paper we incorporate the pathways into the Where-What-Network, modeling the effect of reward and punishment in the dorsal pathway and ventral pathway. Inspired by the instructional scaffolding process proposed by A. N Applebee in [1] when he was observing how young children learns to write, we implement a coarse to fine location concept learning scheme to the network which allows the network to learn finer location concepts using previously learned locations. We formalize the definition of teaching cost and prove the efficiency of integrated training scheme. We also demonstrate the robustness of the network under several mixed training schemes: (1) small percentages of errors in instructions when learning is supervised, and (2) small percentages of errors in reinforcement when using integrated learning training scheme. This shows the temporal and spatial optimality of the internal brain function, which is constructed based on the Lobe Component Analysis proposed by M. Luciw and J. Weng in [16].

Zejia Zheng and Juyang Weng are with Michigan State University, East Lansing, MI, USA (email zhengzej, weng@cse.msu.edu). Juyang Weng is also with the MSU Cognitive Science Program and the MSU Neuroscience Program. Zhengyou Zhang is with Microsoft Research, Redmond, WA, USA.
The authors would like to thank Steve Paslaski and Yuekai Wang for their aid in providing the source code for their networks.

A. Novelty and importance

The approach in this paper is novel in several ways:

1) Integrated Learning: We integrate the reward and punishment learning pathways into Where What Network, which originally uses supervised learning scheme. The network now can be trained with supervised learning and reinforcement learning algorithm, which is one step closer to mimic the learning procedure of human infants. Our previous works in reinforcement learning only implement these two pathways in Developmental Network, which only has one concept zone. WWN is much more complicated because two concepts need to be learned separately.

Integrated learning systems has been found in Actor-critic architecture [10], CLARION [13], and Unified Learning Paradigm [2]. However, the first model is heavily based on TD algorithms, which uses hand crafted computation to deal with reinforcement. The last two models separate their computation with different modules, which is not biologically plausible. Likewise, the feed forward neural networks in [4] introduce Q-learning into the network architecture, which is different from our approach. The integrated system introduced in this paper uses the same neural computation regardless of the availability of supervision or reinforcement. It functions as a developmental program and fully integrates reinforcement learning using neuromodulation.

2) Coarse-to-fine learning: We use a coarse to fine learning scheme, inspired by the concept of instructional scaffolding as is explained in section I-D, for the network to learn spatial concepts. This allows the network to use the already learned concepts to learn finer spatial concepts. This also minimizes the number of educated guesses when the network is trying to figure out the correct action. We formally prove that coarse-to-fine learning minimizes teaching cost compared to supervised

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

learning and reinforcement learning.

Similar concept is proposed by J.L.Elman in [3]. J.L.Elman's network grows in size as the task is proven to be beyond the ability of current network. In our work, however, we scaffold by refining the motor area instead of changing the network architecture. Splitting motor neurons to refine the learned skills is novel as far as we are aware.

B. Where What Network

Where What Networks [5] are a visuomotor version of the Developmental Network, modeling the dorsal (where) stream and the ventral (what) stream of visual and behavioral processing.

Where-What Networks (WWN) have been successfully trained to perform a number of tasks such as visual attention and recognition from complex backgrounds [7], stereo vision without explicit feature matching to generate disparity outputs [11], and early language acquisition and language-based generalization [8].

The lobe component analysis (LCA) [16] is used as an algorithm for neural learning in a cortical area in WWN. It uses the Hebbian mechanism to enable each neuron to learn based on the presynaptic and post-synaptic activities that are locally available to each synapse. In other words, the learning and operation of WWN do not require a central controller.

The learning algorithm and the network architecture are introduced in detail in section II-A.

C. Neuromodulation: reward and punishment

The Motivated Developmental Network uses biological plausible neural computation methods to form the optimal feature representation in the Y area [17]. In the previous work, we built a motivational system that learns the reward and punishment not only by excitation and inhibition in the action area (done by S. Paslaski & C. VanDam [9]), but also by extracting the representation of important events based on neuromodulation. The dynamic learning rate allows the network to form spatial and temporal optimal representation over the important events when the input data set contains irrelevant information.

The model consists of three pathways: an unbiased pathway, a dopamine pathway to learn reward and a punishment pathway to learn punishment. When the event is important, which is defined as rewarded or punished when the agent choose an action, the VTA or RN would release dopamine or serotonin based on the reinforcement. Corresponding pathway would then link the event, the action and the reinforcement together. The learning rate of the network would also be much higher thus allocating more resources to learn the event. Reward pathway would excite certain favorable behaviors and the punishment pathway would inhibit certain behaviors leading to punishment.

D. Coarse-to-fine learning

Coarse-to-fine learning is a learning process designed to promote a deeper level of understanding using previously learned knowledge. The concept has its origins in the work of the psychologist Vygotsky as well as in studies of early language learning [1].

Coarse-to-fine learning is desirable because it allows reinforcement learning in complicated tasks. The neuromodulation approach in this paper requires the network to make educated guesses before learning. The guess however, would turn out to be totally irrelevant if the task is too complicated. As we explain later in section III, the cost would increase quadratically with respect to the task complexity. Coarse-to-fine learning however, reduces the growth rate into a linear function,allowing implementation on complicated tasks.

II. Network Architecture

A. Where What Network

Here we introduce how learning takes place in Where-What-Network.

As illustrated in Figure 1, the network consists of one area of neurons modeling the early sensory areas LGN/V1/V2. The signals then diverge into two pathways, the dorsal (or where) pathway, and the ventral (or what) pathway. The two pathways are bidirectionally connected to the location area and the type area in the frontal cortex, respectively. Unlike the sensory cortex, we assume that the outputs from the location area and the type area can be observed and supervised by teachers (e.g., via the motor areas in the frontal cortex).

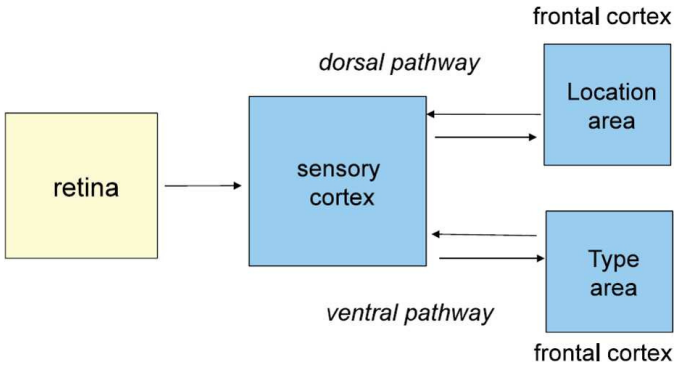

Fig. 1. A schematic of the Where-What Networks (WWN). It consists of a sensory cortex which is connected to the What area in the ventral pathway and to the Where area in the in the dorsal pathway.

B. The learning algorithm

The learning algorithm of WWN is described in detail in previous papers [6] [12] [14]. Here we only introduce the main points of the algorithm to avoid redundancy.

1) Pre-response of the neurons: Each brain area uses the same area function f, which allows the internal area to develop representations of the training data. Generally speaking, each neuron in internal area stores two sets of weight vector $(\mathbf{v}_b.\mathbf{v}_t)$, representing bottom-up weight and top-down weight separately. Similarly, neurons in motor area only have bottom-up weight, while neurons in sensors only have top-down weight. The top-down weight in sensors is useful when we need to predict future images based on current internal

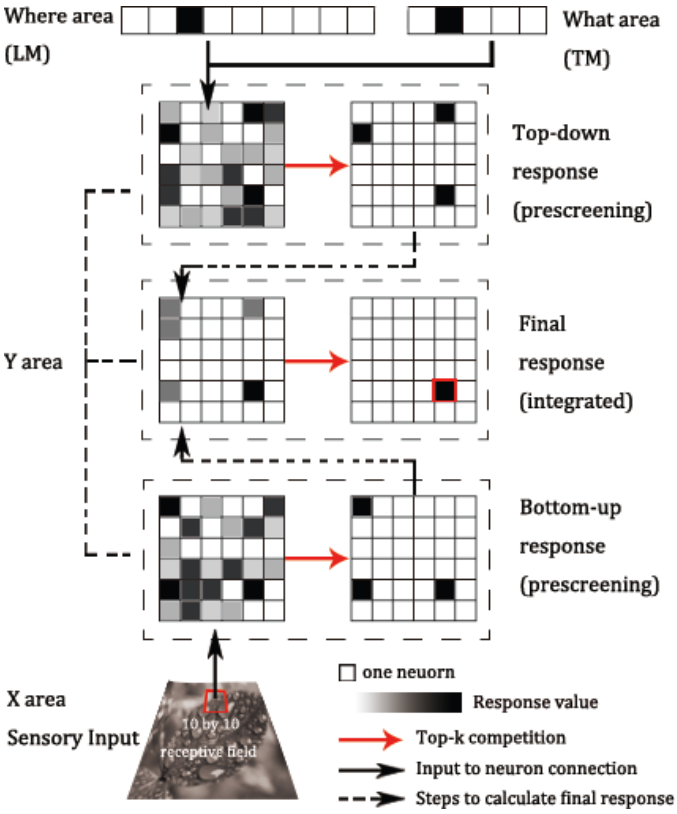

Fig. 2. An example of activation patterns and neuronal changes during learning process in the network. The network selects winning neuron based on top-k competition over the top-down responses and bottom-up responses. The pattern on the left shows the preresponse of the neuron, and the grid on the right shows the final response of the neuron. The winning neuron is marked in red at the final response layer.

responses. In the current program we do not need that set of weight.

The pre-response value for each neuron is calculated as:

$$r(\dot{\mathbf{v}}_b, \dot{\mathbf{b}}, \dot{\mathbf{v}}_t, \dot{\mathbf{t}}) = (\dot{\mathbf{v}}_b \cdot \dot{\mathbf{b}} + \dot{\mathbf{v}}_t \cdot \dot{\mathbf{t}})/2 \quad (1)$$

where $\mathbf{b}$ and $\mathbf{t}$ are bottom up input and top down input respectively. Each vector in equation (1) is normalized before calculation:

$$\dot{\mathbf{v}} = \mathbf{v}/||\mathbf{v}||$$

Each neuron in the Y area extracts local input from the input image. The local window is called *receptive field* of that neuron, depicted in Fig. 2 as the red box in the input image.

Neurons in the Z area accepts the global response values of all the neurons in the Y area as bottom up input. The response values are calculated based on top-k competition explained in the following subsection.

2) Top-k competition: The final neuron response in Y area is given by top-k competition. The k neurons with the highest pre-response value would fire with the adjusted responses while other neurons would be suppressed. To adjust the response values based on their ranking:

$$r' = \begin{cases} r \cdot (r - r_{k+1})/(r_1 - r_{k+1}) & r_1 \le r \le r_{k+1} \\ 0 & otherwise \end{cases}$$

where r_1 is the highest response value; r_{k+1} is the $k+1$th highest response value.

3) Hebbian-like learning: If a neuron wins in the multistep lateral competitions described above, its bottom up weight and top down weight would update using the following Hebbian learning rule:

$$\mathbf{w}_{u,i} \leftarrow \beta_1 \mathbf{w}_{u,i} + \beta_2 r_i \mathbf{x}_t$$

where $\beta 1$ and $\beta 2$ determine retention and learning rate of the neuron, respectively:

$$\beta_1 = \frac{m_i - 1 - \mu(m_i)}{m_i}, \beta_2 = \frac{1 + \mu(m_i)}{m_i} \quad (2)$$

with $\beta_1 + \beta_2 \equiv 1$, m_i is the neuron's firing age, i.e. $m_i = 1$ in the beginning of training, and increments by one every time the neuron wins lateral competition.

μ is a monotonically increasing function of m_i that prevents the learning rate β_2 from converging to zero as m_i increases.

$$\mu(m_i) = \begin{cases} 0, & \text{if } m_i < t_i \\ c(m_i - t_1)/(t_2 - t_1), & \text{if } t_1 < m_i < t_2 \\ c + (t - t_2)/\gamma, & m_i > t_2 \end{cases}$$

We used typical value $t1 = 10, t2 = 10^3, c = 1 and \gamma = 10^4$ in the experiment.

The same Hebbian learning rule updates the top-down weights of neuorns using similar equation:

$$\mathbf{w}_{d,i} \leftarrow \beta_1 \mathbf{w}_{d,i} + \beta_2 r_i \mathbf{r}_t$$

The firing Z neuron accepts Y area firing patterns as bottom up input and updates using the same Hebbian learning rule.

C. Motor neuron splitting: use old knowledge to learn new skills

When a young child learns to recognize the object at certain location, he learns the spatial concepts gradually. He also uses the already learned knowledge to help him acquire new skills. [15] shows how children develop projective/Euclidean understanding before they could comprehend *in front of* and *behind*.

Our network currently models this procedure in the location motor (LM), but similar procedures can be applied to type motor (TM) easily. Coarse to fine learning is achieved by splitting each of the location motor neuron into four child neurons, each representing a finer spatial concept, as is illustrated in Fig.3.

The following steps would take place when LM splitting occurs:

1) Child location motor neuron copies $\mathbf{v}_b$ from parent neuron.
2) Firing age of child location motor neuron set to 1 (or a very low number).
3) Y area neurons copy connection to the parent neuron $\mathbf{v}_t$ to the child location motor neuron.

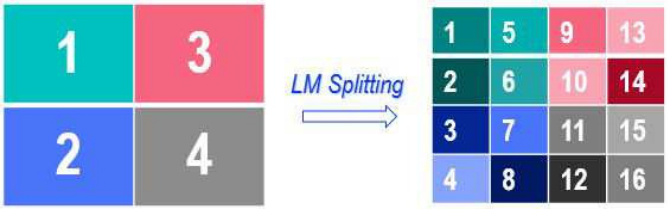

Fig. 3. LM splitting mechanism. Each location motor neuron splits into four child neurons to learn finer spatial concepts.

D. Reward and punishment pathways

The architecture of the two pathways are introduced in detail in the previous papers [9] [17]. Here we only give a brief summary due to limited space.

Previous work modeled two neurotransmitters in human brain: serotonin and dopamine. These two neurotransmitters are released separately in rewarded or punished events. The model simplifies the role of those two neurotransmitters, i.e. dopamine is released when the agent is rewarded, and serotonin is released when the agent is punished. The papers demonstrated that the reward and punishment system built based on Developmental Network enables the agent to learn according to the sweetness(reward) and pain(punishment) it receives when making educated guesses rather than specific instructions of correct movement(supervision).

Although Fig.II-D has 11 areas on the plot, in the program we simplify that architecture into three pathways: $X_u \rightleftharpoons Y_u \rightleftharpoons Z_u$ is the unbiased pathway, $\{X_u, X_p\} \rightleftharpoons Y_p \rightleftharpoons Z_p$ is the punishment pathway, and $\{X_u, X_p\} \rightleftharpoons Y_s \rightleftharpoons Z_s$ is the reward pathway.

VTA and RN are treated as conceptual area that trigger firing when reward or punishment is present, corresponding to two *if* clauses in the program. Y_{VTA} and Y_{RN} are the same areas as Y_p and Y_s, using different neuromodulators.

The network calculates three responses: $\mathbf{r}_u$ for response value in the unbiased pathway, $\mathbf{r}_p$ for response value in the punishment pathway and $\mathbf{r}_s$ for response value in the reward pathway.

The final response value is given by:

$$r_i \leftarrow r_{iu}(1 + r_{is} - \gamma r_{ip}) \tag{3}$$

γ is usually larger than 1, indicating that inhibition from the pain pathway is much more effective compared to excitation from the reward pathway.

Another effect of neuromodulators is that they would increase the learning rate in the corresponding areas. This would change equation 2:

$$\beta_1 = 1 - \beta_2, \beta_2 = \alpha \cdot \frac{1 + \mu(m_i)}{m_i} \tag{4}$$

In the experiment, $\alpha = 2$.

Where What Network, being an upgraded version of Developmental Network with a dorsal stream and a ventral stream, should be able to incorporate the reinforcement learning schedule as well. The difficulty, however, lies in the enormous amount of educated guesses generated by the combination of two Z areas (5 types and 64 locations generate 320 possible guess result). This is no longer a problem in the incremental skill learning scheme. A new location, which corresponds to a new neuron in Z area, inherits its bottom up weight and other properties from its parent neuron, which corresponds to a larger area enclosing the target location. Assuming the agent learns the old set of location well enough, then the agent would just need to distinguish the child neurons of one specific parent neuron.

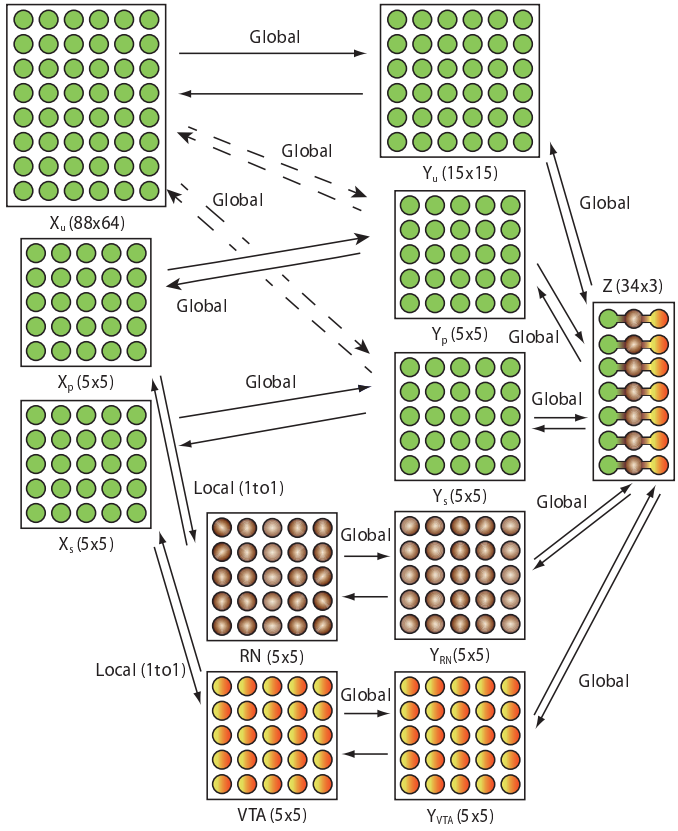

Fig. 4. A DN with 5-HT and DA modulatory subsystems. It has 11 areas. RN has serotonergic neurons. Neurons in Y_{RN} or Z have serotoninergic synapses. VTA has dopaminergic neurons. Neurons in Y_{VTA} or Z have dopaminergic synapses. The areas $Y_u, Y_p, Y_s, Y_{RN}, Y_{VTA}$ should reside in the same cortical areas, each represented by a different type of neurons, with different neuronal densities. Within-area connections are simulated by top-k competition.The number of neurons in the figure does not necessarily corresponds to the number of neurons we use in the experiment. The figure does not show all the neurons in the subareas due to limited space. The Z neurons compute unbiased response(the first column, using glutamate), inhibition response (second column, using serotonin), and excitation response (third column, using dopamine). "Global" indicates that the neurons accept input from all neurons in the previous area. RN (raphe nuclei) is the subarea inside human brain that secretes serotonin. VTA (Ventral tegmental area) is the subarea that secretes dopamine. Basically Xp and Xs does not produce those neurotransmitters but only sends input signals to those two brain areas. The function of these two layers is for biological correctness. Thus each neuron in X_p (or X_s) corresponds to one neuron in RN (or VTA). The link is one to one, thus "Local".

III. Analysis on teaching cost

The goal of this integration system is to reduce the cost of teaching by blending reinforcement learning and supervised learning together. We are going to prove in this section that coarse to fine learning process would reduce cost in teaching.

A. Definition

To justify the claim we need to formalize the definition of cost of teaching. The intuition is that the cost of supervision, when the task is complicated, should be higher compared to the cost of given reward and punishment. Supervising the act of the agent would require the teacher to give detailed instruction to the learner, while giving reward and punishment is much easier for the instructor. Thus we define the cost of teaching as follows:

1) Cost of supervision. As is analyzed above, the total cost of supervision is proportional to the cost in each supervision and the number of instructions given. Thus we have:
$$C_s = n_s \mu_s \tag{5}$$
where μ_s denotes the cost in each detailed instruction, and n_s is the number of instructions. μ_s remains as a constant because the teacher is assumed to have already mastered the task and therefore can offer instruction at constant cost.
2) Cost of giving reward and punishment. To give reward/punishment, the teacher has to compare the educated guess given by the agent with the correct movement. The cost of the comparison remains constant because the teacher already knows the answer. Thus the cost C_r to give reward or punishment is defines as:
$$C_r = n_r \mu_r \tag{6}$$
where n_r is the number of reward/punishment given during learning. μ_r is usually much smaller than μ_s because giving detailed supervision would require much more elaborate instruction.
3) Total cost. Total cost of the teaching process is easy once we define the two costs above.
$$C = C_s + C_r \tag{7}$$

One thing to notice about the above definitions is that these definitions are based on intuition and may suffer from inaccuracy due to lack of supporting literature. On the other hand we found these definitions useful as a guidance in our integrated system and these definitions are in accordance with our daily experience.

B. Cost Analysis

Here we compare teaching cost in three different approaches using the network architecture we introduced in previous sections.

The tasks is to recognize n_t types of objects, which would appear at n_l locations in the input image. For simplicity, let us assume that n_l is the power of 4, supervised learning and reinforcement learning takes k epochs, meaning that we train each object at each location twice.

1) Supervised Learning: From eq. (5) and (7), total cost can be calculated as:
$$\begin{aligned} C &= n_s \mu_s \\ &= k n_t n_l \mu_s \end{aligned} \tag{8}$$

2) Reinforcement Learning: Reinforcement learning is a little bit complicated. The network has to make educated guesses each time. Assuming we start from scratch (random weight) when training the network, the expected number of guesses n_g at each given image is
$$E(n_g) = n_t n_l / 2 \tag{9}$$
This is because the network has not learned anything for the first epoch, so it can make all possible guesses with equal probability. After the first epoch, however, the reward pathway links the reward with the input and its correspond correct action, thus ideally the network would always perform correctly.

Thus the expected total cost can be calculated as:
$$E(C) = E(C_r) = \mu_r (E(n_r^1) + \sum_{i=2}^{k-1} E(n_r^i)) \tag{10}$$
where n_r^i stands for the number of reward/punishment given in the ith epoch. Based on previous discussion, $n_r^1 = n_t n_l E(n_g) = (n_t n_l)^2/2$ and $n_r^i = n_t n_l$, for $i > 1$.

Following eq.(10), we have:
$$\begin{aligned} E(C) &= \mu_r ((n_t n_l)^2/2 + (k-1) n_t n_l) \\ &= \mu_r n_t n_l (n_t n_l / 2 + k - 1) \end{aligned} \tag{11}$$

3) Integrated Learning: With only one split in LM motor, the integrated learning process can be separated into 3 steps:

1) learn $n_l/4$ locations and n_t types with supervision. The number of supervision is lowered by 4 because we are training at less locations. This step would take k epochs and the cost C_s would be $C_s = k \cdot \mu_s \cdot n_l \cdot n_t / 4$.
2) split LM neuron as is introduced in previous section.
3) learn n_l locations and n_t types with reinforcement for k epochs. The difference now is that the expected number of guesses at the first epoch would no longer be $n_t n_l / 2$ as is in 9. Based on previous experience, the network has learned to narrow the possible location into the four child locations. The expected number of guesses is now cut down to 2 in ideal cases. Thus we have:
$$\begin{aligned} E(C_r) &= \mu_r (E(n_r^1) + \textstyle\sum_{i=2}^{k-1} E(n_r^i)) \\ &= \mu_r (2 n_t n_l + + (k-1) n_t n_l) \\ &= \mu_r n_t n_l (k+1) \end{aligned}$$

Putting all steps together, we have,
$$\begin{aligned} E(C_i) &= C_s + E(C_r) \\ &= k \mu_s n_t n_l / 4 + \mu_r n_t n_l (k+1) \\ &= n_t n_l (k \mu_s / 4 + \mu_r (k+1)) \end{aligned} \tag{12}$$

TABLE I
TEACHING COST IN DIFFERENT LEARNING SCHEMES

Learning Scheme	Expected Cost
supervised	$k\mu_s n_t n_l$
reinforcement	$\mu_r(n_t n_l/2 + k - 1)n_t n_l$
integrated	$(k\mu_s/4 + \mu_r(k+1))n_t n_l$

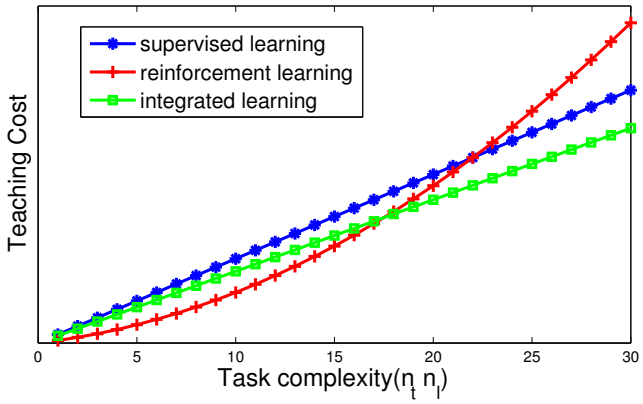

Fig. 5. Teaching cost of three different learning schemes. Plot parameter: $\mu_s = 3, \mu_r = 1, k = 3$. This plot is based on table I. Reinforcement learning becomes more costly as the task complexity increases. The cost of supervised learning and integrated learning remains proportional to the complexity. Moreover, integrated learning cost less compared to supervised learning.

4) Comparison and Discussion: The expected cost of all three learning schemes is summarized in table I. Fig.5 illustrates how the cost changes with respect to the complexity of the given task. For simple tasks, reinforcement learning would cost less because there is limited number of educated guesses. However, as the complexity increases (more locations/types to learn), reinforcement learning would not be plausible because the cost would increase quadratically instead linearly with respect of $n_t n_l$.

The cost of integrated learning, on the other hand, remains linear with respect to the complexity of the given task. It is also lower than the cost of supervised learning, which makes it a desirable learning scheme. One thing to notice is that the LM neuron only splits once in our analysis. We anticipate the difference would be more noticeable with more splitting in the LM neurons.

IV. EXPERIMENT

We show that the performance of our integrated learning system in this section.

A. Coarse to Fine Learning

The network is first trained to learn rough locations of the foreground object. During the first epoch we train only four different locations: upper left, upper right, lower left and lower right. The network architecture then splits each one of its motor neurons into four neurons to learn to discriminate in higher precision. The new neurons copy the weights and connections of its parent neuron. The four new motor neurons represents

Fig. 6. Sample training and testing images. Training images are generated by stitching foreground objects onto different locations in the random natural background images. In the experiments we used five different types of foreground object: cat, dog, truck, pig and elephant.

four sub-locations of the parent neuron. The network then goes through training process once again to refine those copied neurons. More splitting and training would take place if higher precision is required.

This is the base case of our experiment. Training is achieved using supervised training scheme described in previous sections. The network learns to recognize 5 foreground objects at 16 places and 64 places.

Result is shown in table II.

Data: training object list, training location list
Result: foreground object type and location initialization;
% Training phase starts ;
for *each training object* **do**
  for *each true_location* **do**
    Generate training picture;
    Get rough location;
    Find firing neuron(object type, rough location, training picture);
    Update $\mathbf{v}_b$, $\mathbf{v}_t$ of firing neuron;
    Update $\mathbf{v}_b$ of firing z neuron in LM and TM;
  end
end

Algorithm 1: Supervised learning environment

B. Supervised learning with errors in instruction

When trained in real time, the network would occasionally receive mislabeled training data. Instead of trying to handcraft a filter to get rid of these "bad" data beforehand, our network remembers those mislabeled data and tries to eliminate those inconsistent labels via top-k competition.

In the experiment, we added in small portions of error in labels when training the network. As one can easily expect, recognition rate is lower with more errors in the training data.

C. Reinforcement learning

In this experiment we added in reinforcement learning mechanism into the previous network setup. Two more pathways are added to the network as is described in section II-D

The first four rough locations are taught to the network using supervised training scheme. The network then splits its motor neurons to learn the foreground object in higher precision.

For a given input image, the network should be able to recall its rough location, which corresponds to the spatial concept

```
Data: training object list, training location list
Result: foreground object type and location
initialization;
% Training phase starts ;
for each training object do
    for each true_location do
        Generate training picture;
        Get rough location;
        while guess num ≤ max guess num do
            make educated guesses;
            if guess wrong then
                Give punishment;
            else
                Give reward;
            end
            if reward is delayed then
                while delay do
                    Learn current behavior;
                end
            end
            Learn reward or punishment;
        end
    end
end
```

Algorithm 2: Reinforcement learning environment

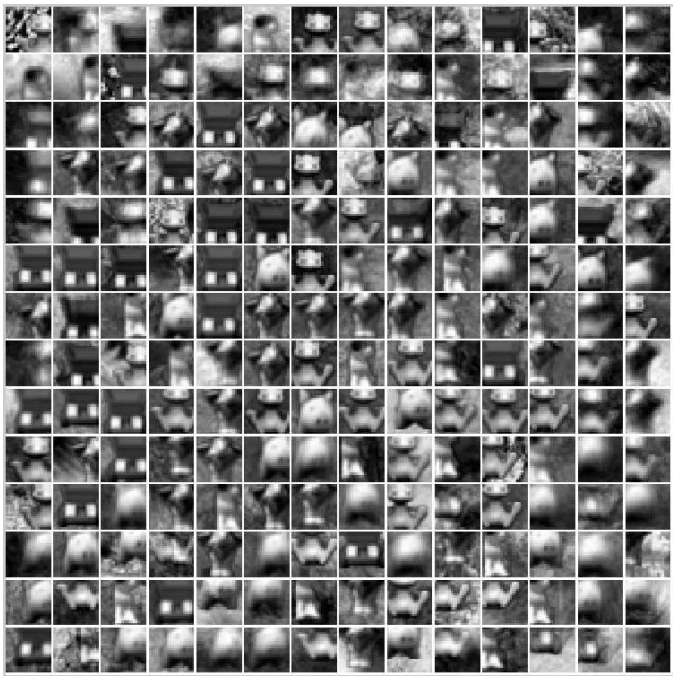

Fig. 7. Bottom-up weight in Y neurons. This picture shows the bottom up weight of the Y area neurons, layer 1, after splitting 2 times and trained with reinforcement, 9 epochs at each split. 5 more similar layers are used in the network. The picture shows that the network successfully captures the features of the foreground objects following algorithm 3.

```
Data: training object list, training location list
Result: foreground object type and location
initialization;
% Training phase starts ;
Initialize network with 4 LM neurons ;
Train network supervised at 4 rough locations;
while LM neuron num ≤ target num do
    split LM neurons;
    while epoch num leq max epoch num do
        if epoch num == 1 then
            train network with supervision;
        else
            train network with reinforcement;
        end
    end
end
```

Algorithm 3: Integrated learning environment

TABLE II
SUPERVISED LEARNING WITH ERROR IN INSTRUCTION

TM error rate (%)	LM distance (pixels)	Error in instruction (%)
0	0.4595	0
3.44	0.6865	5
4.37	0.7892	10
6.25	1.0865	15

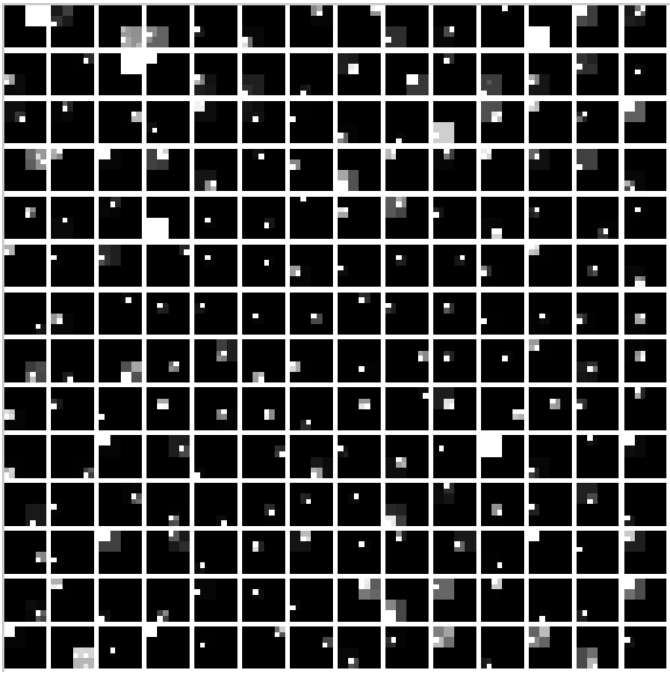

Fig. 8. Top-down LM weight for Y neurons. This picture shows the top-down weight for Y neurons in the first layer, after splitting 2 times and trained with reinforcement, 9 epochs at each split. 5 more similar layers are used in the network. The picture shows that the network successfully linked most neuron with one corresponding location after training following algorithm 3.

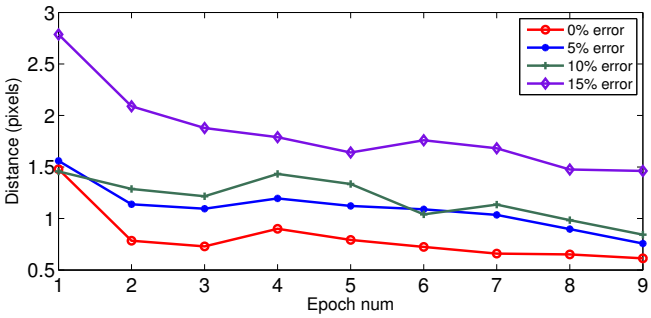

Fig. 9. LM average distance with error in reinforcement. The figure shows the distance between the recognized location and the true location with different percentages of error in reinforcement. The figure shows that the network can tolerate small percentages of wrong instructions.

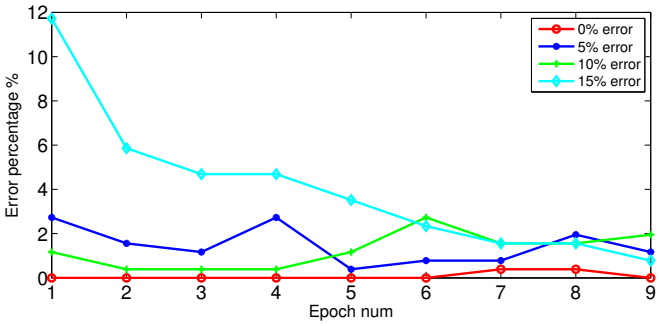

Fig. 10. TM error rate with error in reinforcement. The figure shows the error percentage of the type recognition result with different percentages of error in reinforcement. The figure shows that the network can tolerate small percentages of wrong instructions.

prior to splitting. This means that after splitting, only 4 neurons in the new Z area would share the highest response value. Thus the network would make at most four guesses before it receives a reward for correct behavior. This approach minimizes the number of educated guesses needed for the network to figure out the correct answer.

In the experiment, the network is allowed to make at most 10 educated guesses at each picture.

D. Errors in reinforcement

As is explained in IV-B, training data may contain errors. We added errors in the reward and punishment to further exploit the network's potential. The wrong reinforcement in the experiment is issued as follows: a) if the educated guess is correct, then give punishment at both pain sensor for location motors and pain sensor for type motors. b)If the educated guess is wrong, however, reward is given to the network. Recognition rate in TM is plotted in Fig.10. Location recognition distance is plotted in Fig.9.

V. Conclusions

This paper extends the previous work on Where-What-Network in a new direction- integrated learning. In the new learning mode, both the agent and the environment are allowed to alter the mode of learning(supervised or reinforcement learning). Our analysis showed that integrated learning have a lower cost compared to purely supervised or purely reinforcement learning modes. The quality of either learning mode affects the overall speed of learning.

References

[1] Arthur N Applebee and Judith A Langer. Instructional scaffolding: Reading and writing as natural language activities. *Language arts*, 60(2):168–75, 1983.

[2] Edward Y Chang, SCH Hop, Xinjing Wang, Wei-Ying Max, and Michael R Lyu. A unified learning paradigm for large-scale personalized information management. In *Emerging Information Technology Conference, 2005.*, pages 4–pp. IEEE, 2005.

[3] Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

[4] Bing-Qiang Huang, Guang-Yi Cao, and Min Guo. Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 1, pages 85–89. IEEE, 2005.

[5] Zhengping Ji, Juyang Weng, and Danil Prokhorov. Where-what network 1:where and what assist each other through top-down connections. In *Development and Learning, 2008. ICDL 2008. 7th IEEE International Conference on*, pages 61–66. IEEE, 2008.

[6] Matthew Luciw and Juyang Weng. Where-what network 3: Developmental top-down attention for multiple foregrounds and complex backgrounds. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

[7] Matthew Luciw and Juyang Weng. Where what network 3: Developmental top-down attention with multiple meaningful foregrounds. In *International Joint Conference on Neural Networks*, pages 4233–4240, 2010.

[8] Kajal Miyan and Juyang Weng. Wwn-text: Cortex-like language acquisition with what and where. In *Development and Learning (ICDL), 2010 IEEE 9th International Conference on*, pages 280–285. IEEE, 2010.

[9] Stephen Paslaski, Courtland VanDam, and Juyang Weng. Modeling dopamine and serotonin systems in a visual recognition network. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 3016–3023. IEEE, 2011.

[10] Michael T Rosenstein, Andrew G Barto, Jennie Si, Andy Barto, Warren Powell, and Donald Wunsch. Supervised actor-critic reinforcement learning. *Handbook of Learning and Approximate Dynamic Programming*, pages 359–380, 2004.

[11] Mojtaba Solgi and Juyang Weng. Developmental stereo: Emergence of disparity preference in models of the visual cortex. *Autonomous Mental Development, IEEE Transactions on*, 1(4):238–252, 2009.

[12] Xiaoying Song, Wenqiang Zhang, and Juyang Weng. Where-what network 5: Dealing with scales for objects in complex backgrounds. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2795–2802. IEEE, 2011.

[13] Ron Sun. The clarion cognitive architecture: Extending cognitive modeling to social simulation. *Cognition and multi-agent interaction*, pages 79–99, 2006.

[14] Yuekai Wang, Xiaofeng Wu, and Juyang Weng. Skull-closed autonomous development: Wwn-6 using natural video. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE, 2012.

[15] Susan Wanska. The relationship of spatial concept development to the acquisition of locative understanding. *The Journal of genetic psychology*, 145(1):11–21, 1984.

[16] Juyang Weng and Matt Luciw. Dually optimal neuronal layers: Lobe component analysis. *IEEE Transactions on Autonomous Mental Development*, 1(1):68–85, 2009.

[17] Zejia Zheng, Qian Kui, Juyang Weng, and Zhengyou Zhang. Modeling the effects of neuromodulation on internal brain areas: Serotonin and dopamine. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013.